# A2 COMPUTER SCIENCE PAPER 2 MOCK EXAM 11/04/19

## Algorithms and Programming

Name: _____

Mark: /140

Grade:

APRIL 11, 2019

**Name: _____** **Mark:** **/140** **Grade:**

**Section A**
Answer **all** questions.

**1** The map shown below in Fig.1 is of a city and surrounding districts. The map is an *abstraction* made up of a number of component parts.

**Fig.1**

**(a)** Using the map in Fig.1 as an example, define the term *abstraction*.

.....

.....

.....

.....**[4]**

**(b)** Both map making and program development make use of reusable components.

**(i)** Give **three** examples of how reusable component parts are used in Fig.1.

1.....

.....

2.....

.....

3.....

..... [3]

**(ii)** Explain how programmers make use of reusable components when developing large programs.

.....

.....

.....

.....

.....

.....

.....**[3]**

**(c)** Explain **two** advantages to programmers of using reusable components when developing programs.

1.....

.....

.....

.....

.....

.....

2.....

.....

.....

.....

.....

.....

..... **[4]**

**2** Consider the following algorithm in Fig.2, expressed in pseudocode, as a function S:

```
function S(A[0..N-1], value, low, high)

     if (high < low) then
          return error_message
     endif

     mid = (low + high) / 2

     if (A[mid] > value)then
          return S(A, value, low, mid-1)
     elseif (A[mid] < value)then
          return S(A, value, mid+1, high)
     else
          return mid
     endif
endfunction
```

**Fig.2**

**(a)** State the name of the algorithm implemented in Fig.2.

.....**[1]**

**(b)** Describe the purpose of this algorithm.

.....

.....

.....**[2]**

**(c)** Parameters are passed to this function. Complete the following table to identify these parameters and the purpose of each.

| Parameter Name | Purpose |
|---|---|
| | |
| | |
| | |
| | |

[8]

**(d) (i)** Describe what is meant by recursion.

.....

.....

.....

.....**[2]**

**(ii)** Identify **one** example of where recursion occurs in this algorithm.

.....

.....**[1]**

**(e)** Explain how the algorithm in Fig.2 is an example of a divide and conquer approach.

.....

.....

.....

.....

.....
**[3]**

**(f)** Rewrite the algorithm in Fig.2 without using recursion. Annotate your pseudocode with comments to show how it solves the problem.

**[8]**

**3** Julie wants to earn her living by being a successful app developer.
Before she even writes any code, she thinks it would be sensible to find out some basic facts about app development and the market for apps in order to maximise her chances of being successful.

**(a)** State **four** items of data that she could obtain in order to make a sensible choice of an app development project.

1.....

2.....

3.....

4.....
**[4]**

**(b)*** *'It is possible to use computational methods as a way of predicting the success of an app.'*
Discuss the extent to which you agree with this statement.
.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....**[9]**

**4** A supermarket chain stores over 3 petabytes (1015 bytes) of data about sales and customers. The supermarket chain carried out a data mining exercise in which they discovered that whenever there was a hurricane warning, sales of fruit pies increased. This had not been noticed before the data mining exercise. The next time there was a hurricane warning, they placed the fruit pies at the end of the aisles and there was a dramatic increase in sales.
Explain how computational methods were able to reveal this unexpected result.

.....

.....

.....

.....

.....

.....**[4]**

**5 (a)** Describe what is meant by a *heuristic* approach to problem-solving.

.....

.....**[2]**

**(b)** Describe how heuristic methods are used when making a decision about when to cross a busy road.

.....

.....

.....

.....**[3]**

**(c)*** Virus checkers work by looking for patterns in the program code. They also use heuristic approaches. Evaluate the effectiveness of heuristic and pattern matching approaches to virus detection.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....**[9]**

**6** Fig.3 shows a plan of a cafeteria. Customers are complaining that it currently takes too long to collect their drinks.

**Fig.3**

The cafeteria manager is thinking about reorganising the layout of the cafeteria in order to speed up the service to customers.

**(a)** State **three** data items that could usefully be collected in order to investigate the problem.

1.....

2.....

3.....

.....**[3]**

**(b)** Explain how computational methods could be used in order to improve the layout of this cafeteria.

.....

.....

.....

.....

.....

.....

.....**[4]**

**7** When a house is being built, the following activities take place:
- plans are drawn
- foundations are laid
- bricks are ordered
- bricks are delivered
- walls are built
- windows are installed
- electric wiring is installed
- plumbing is installed
- roof rafters are installed
- tiles are put on roof.

**(a)** Describe the term *pipelining*.

.....

.....

.....**[2]**

**(b)** Explain how pipelining principles can be used to ensure that a house is built as quickly as possible.

.....

.....

.....

.....

.....**[4]**

**(c)** Describe **two** examples of where pipelining is used in any computer system.

1.....

.....

.....

2.....

.....

.....

.....**[4]**

**8** It is possible to use XOR together with a key to encrypt a plain text message.

For example, to encrypt the bit pattern 111100001111 using the number 5 as a key, the key is repeated as often as necessary to match the length of the message. An XOR operation is performed to generate the encoded message.

111100001111 <-Message to be encrypted
101101101101 <-Key: repeated as necessary
010001100010 <-Encrypted message

Write an algorithm to accept a text message and a key that would use the method above to generate an encoded message. Annotate your pseudocode with comments to show how it solves the problem.

**[8]**

**9** Linear search and binary search are two different algorithms which can be used for searching arrays.

When comparing linear and binary search it is possible to look at the best, worst and average number of items in the array that need to be checked to find the item being searched for. Assume every item in the array is equally likely to be searched for.

**(a)** Complete the table below

| | Worst case number of searches | Average case | Best Case |
|---|---|---|---|
| Binary Search | | $Log_2(n) - 1$ | |
| Linear Search | | $n/2$ | |
| | | | |

[4]

**(b)** As the size of an array increases the average number of checks grows logarithmically. State what is meant by logarithmic growth.

.....

.....
**[1]**

**(c)** Assuming an array is sorted give a situation when a linear search would perform better than a binary search.

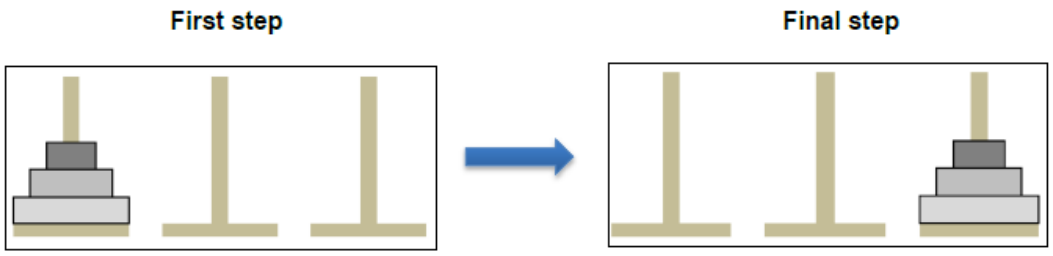
.....

.....
**[1]**

## Section B

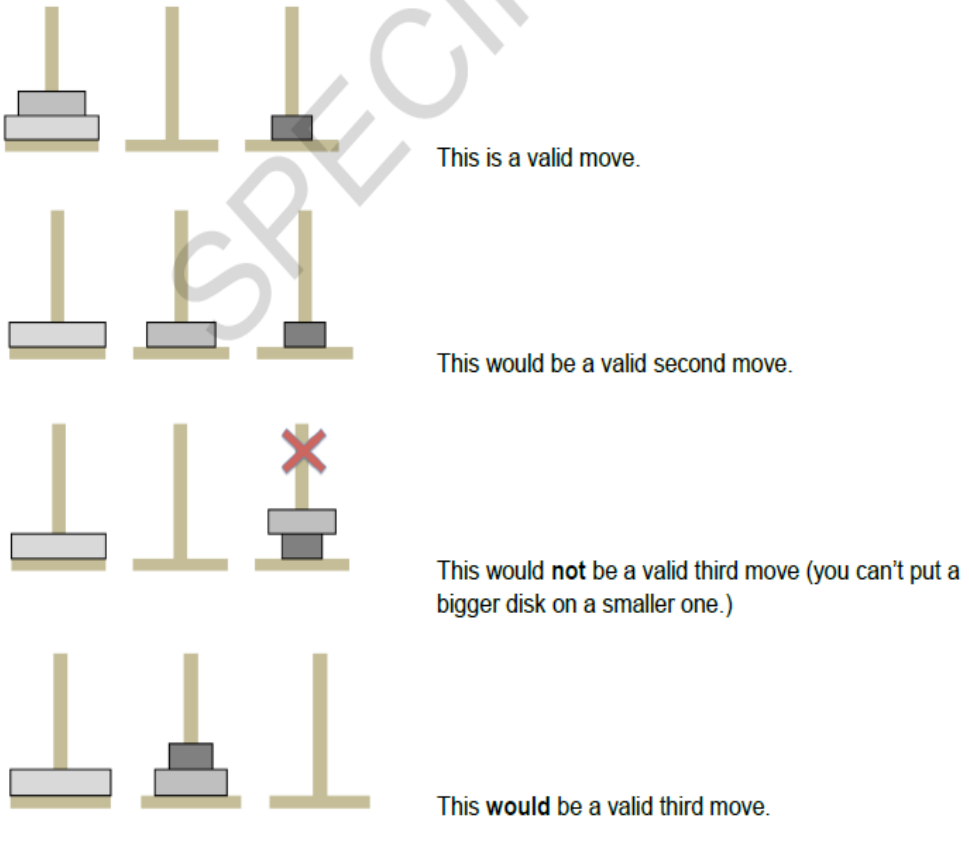
Answer **all** questions.

**10** The Towers of Hanoi is a classic puzzle. Disks are placed in order on a pole, the biggest disc at the bottom of the pole, the smallest disk at the top of the pole, on the first of three poles. The challenge is to get them to the third pole in the same order.

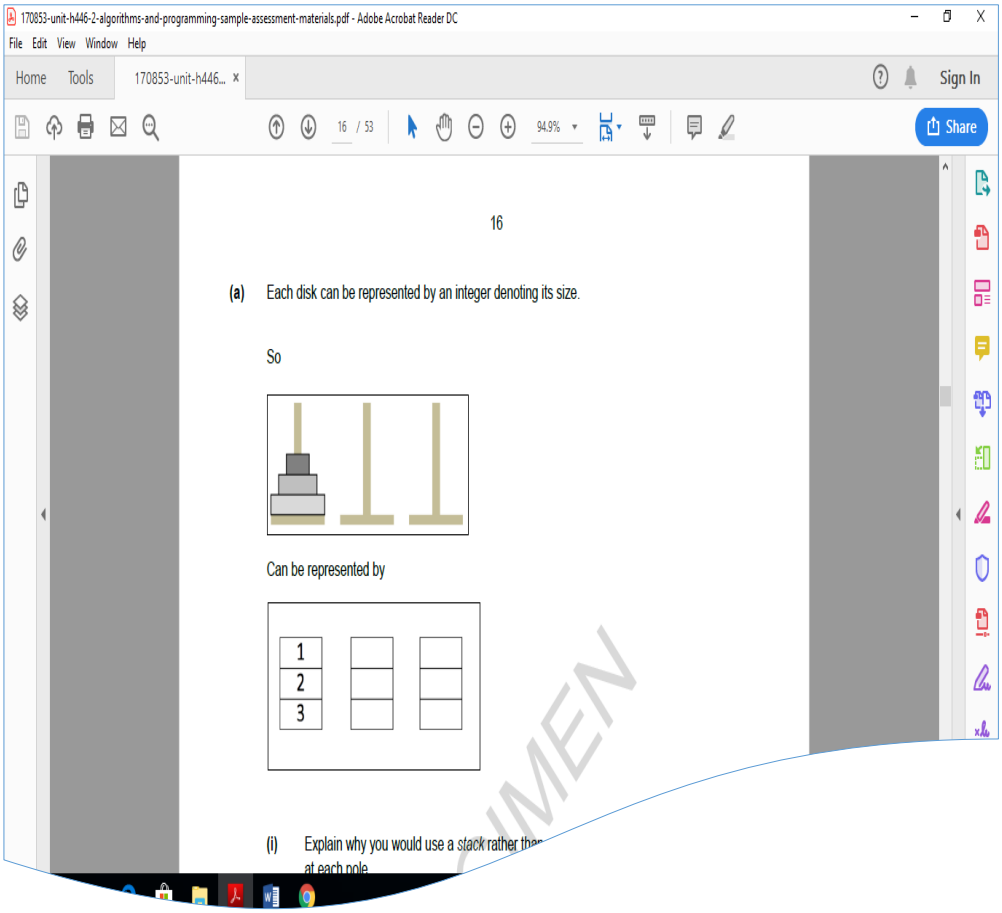



The disks can only be moved under the following rules:

- only one disk can be moved at a time
- a disk can only ever be placed on an empty pole or on top of a larger disk
- a larger disk can never be placed on a smaller disk.


This is a valid move.
This would be a valid second move.
This would **not** be a valid third move (you can't put a bigger disk on a smaller one.)
This **would** be a valid third move.

SPECI




**(i)** Explain why you would use a *stack* rather than a *queue* to store the configuration of disks at each pole.

.....

.....

.....**[2]**

**(ii)** The tower class has the method push. It takes in the value of the disk to be pushed. It adds it to the top of the stack if it is a valid move. If it is not a valid move, the value of the disc is not added and the message 'Invalid move' is printed to the screen.

The stack is implemented using an array called pole and an integer called pointer. Pointer represents the index of the array position at the top of the stack.

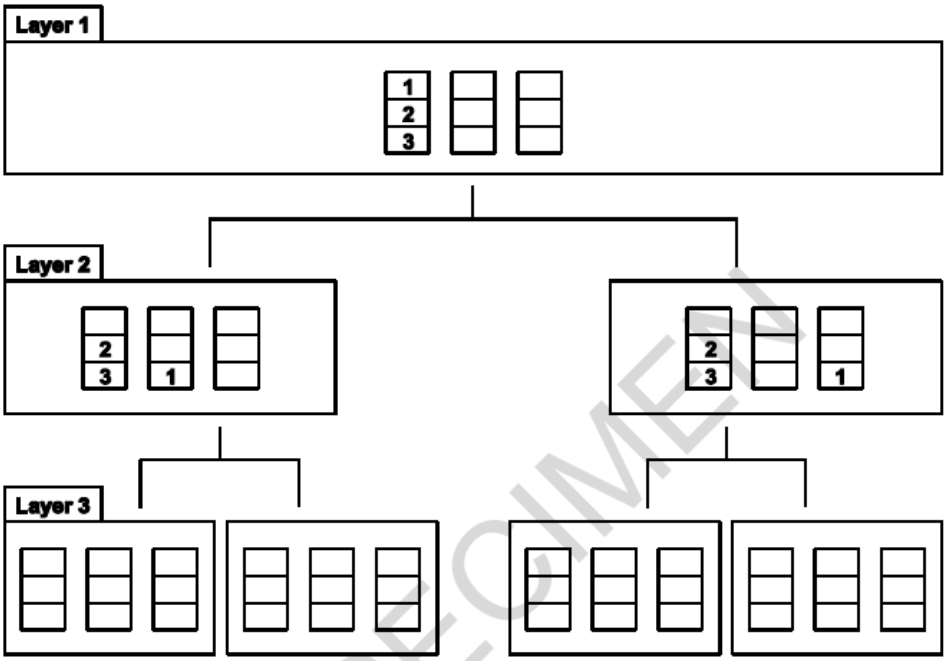
```
class Tower
     private array pole[10]
     private pointer

     public procedure new()
          pointer=0
     endprocedure

     public procedure push(diskValue)
     //Code for push method
     endprocedure

endclass
```

Write the pseudocode to go in the push method. Annotate your pseudocode with comments to show how it solves the problem. You are not expected to test for overflow.

**[6]**

**(b)** One way to try to find a solution would be to generate a tree of possible moves until a solution is found.

**(i)** A tree has been started below. Complete **Layer 3** to show 4 possible moves. **[4]**




SPECIMEN

**(ii)** The search space represented by the tree could be searched using a depth first or breadth first search.

Describe **one** advantage and **one** disadvantage of depth-first search compared with breadth-first search.

Advantage:

.....

.....

.....

.....

Disadvantage:

.....

.....

.....**[4]**

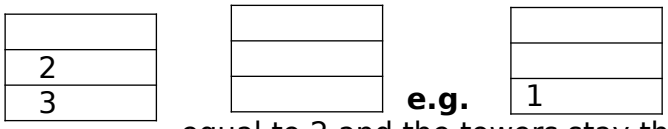
**(c)** Rather than using a tree, the following iterative algorithm can be used to play the perfect game where the number of disks is odd. A similar algorithm exists for an even number of disks.

A program is required to solve the Towers of Hanoi puzzle using the iterative algorithm below.

Iterative algorithm to solve Towers of Hanoi for an odd number of disks.
Cycle through the following three steps until the puzzle is solved (which may be after any of the steps):

- make the valid move between tower1 and tower3
- make the valid move between tower1 and tower2
- make the valid move between tower3 and tower2

NB: The valid move might be in either direction but there will only be one possibility each time.
In this program there are three objects of the class Tower; `tower1, tower2` and `tower3`
The Tower class has the methods `push, peek` and `pop`
The method `push` adds a disk to the tower, for example `tower1.push(3)` adds a disk of size 3 to tower1.
The method `peek` returns the value of the disk on top of the tower but does not remove it. It returns the value 999 if the tower is empty.
The method `pop` removes a disk from a tower and returns the value of that disk.




**e.g.** `x=tower1.peek()` would make x equal to 2 and the towers stay the same.
`x=tower1.pop()` would make x equal to 2 and remove 2 from tower1.

Complete the pseudocode program below so when given an odd number of disks, below 100, on tower1 they will be moved to tower3 using the iterative algorithm.

Annotate your pseudocode with comments to show how it solves the problem. **[10]**

```
noOfDisks=5 //Can be set to any odd number below 100
     tower1 = new Tower()
     tower2 = new Tower()
     tower3 = new Tower()

     i=noOfDisks
     while i>0
          tower1.push(i)
          i=i-1
     endwhile
     //add code to solve puzzle
```

**(d)** The complexity of solving the Towers of Hanoi can be expressed in Big O notation as $O(2^n)$ where n is the number of disks.

**(i)** A given computer takes 8 milliseconds (ms) to solve a 3 disk problem. Calculate how long the computer takes to solve a 5 disk problem.

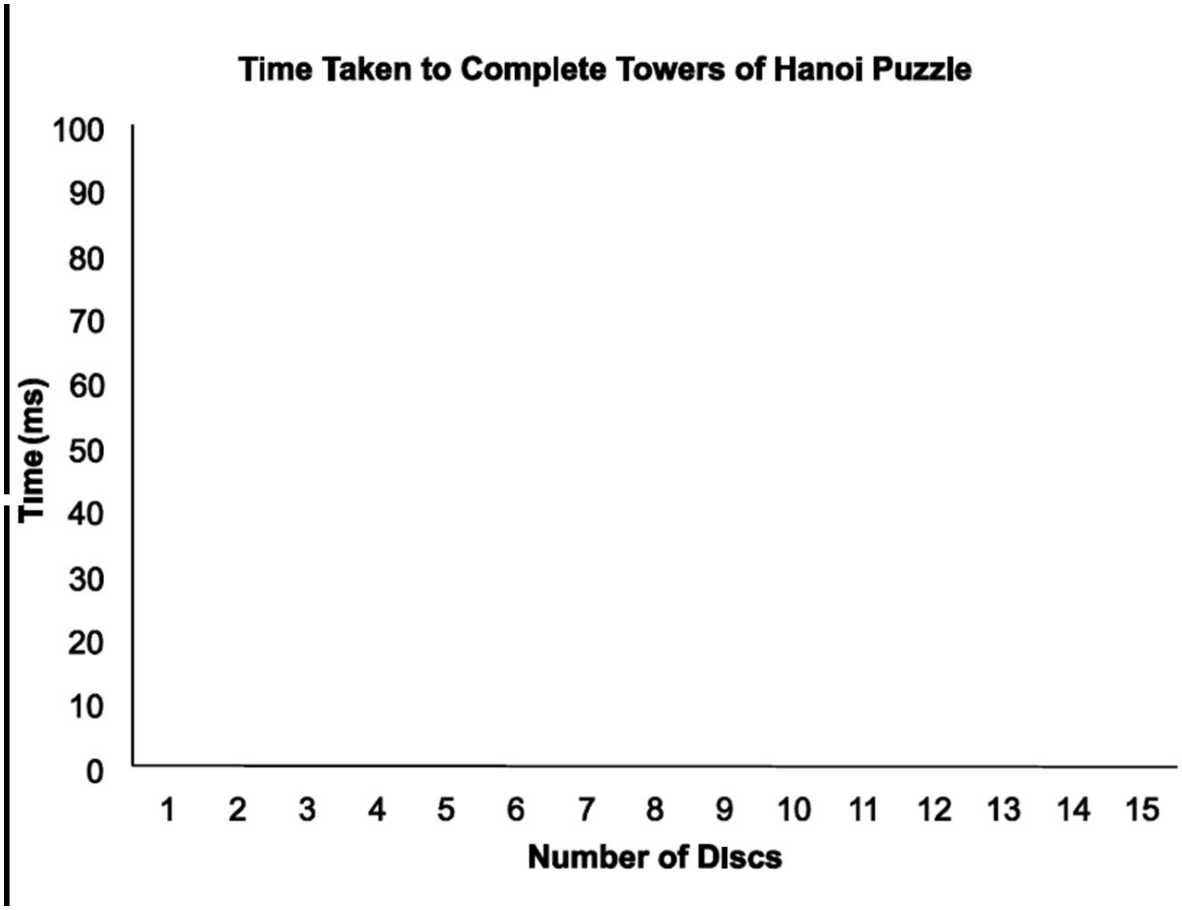
.....
**[1]**

**(ii)** State **one** reason why the answer given for part **(i)** may only be an estimate.

.....

.....
**[1]**

**(iii)** Complete the graph below to show an estimate of how long a computer would take to solve the Towers of Hanoi with a variable number of discs.
**[2]**




**(e)*** Concurrency can be used to speed up the processing of some problems.
Discuss to what extent concurrency might be of use when solving the Towers of Hanoi puzzle. You should consider both the search space tree and iterative algorithm approaches in your answer.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

**[9]**

**END OF PAPER**